#24/Appeal Brief
T. McClellan...
EV188390460 11/18/02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 08/852,158
Filing Date ..... May 6, 1997
Inventor.....Sharad Mathur et al.
Group Art Unit 2151
Examiner ..... Opie, George
Applicant ..... Microsoft Corporation
Attorney's Docket No. MS1-151US
Confirmation No. 6705
Title: Controlling Memory Usage in Systems Having Limited Physical Memory

APPEAL BRIEF

To: Commissioner of Patents and Trademarks
Washington, D.C. 20231

From: Allan Sponseller (Tel. 509-324-9256x215; Fax 509-323-8979)
Customer No. 22801

Pursuant to 37 C.F.R. § 1.192, Applicant hereby submits an appeal brief for application 08/852,158, filed 5/6/97, within the requisite time from the date of filing the Notice of Appeal. Accordingly, Applicant appeals to the Board of Patent Appeals and Interferences seeking review of the Examiner's rejections.

Appeal Brief Items **Page**

(1) Real Party in Interest

The real party in interest is Microsoft Corporation, the assignee of all right, title and interest in and to the subject invention.

(2) Related Appeals and Interferences

Appellant is not aware of any other appeals or interferences which will directly affect, be directly affected by, or otherwise have a bearing on the Board's decision to this pending appeal.

(3) Status of Claims

Claims 1-40 stand rejected and are pending in this Application. Claims 1-40 were previously amended and are set forth in the Appendix of Appealed Claims on page 30.

Claims 1-39 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,835,765 to Matsumoto (hereinafter "Matsumoto") in view of U.S. Patent No. 5,881,284 to Kubo (hereinafter "Kubo").

Claim 40 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,815,702 to Kannan et al. (hereinafter "Kannan") in view of U.S. Patent No. 5,826,082 to Bishop et al. (hereinafter "Bishop").

(4) Status of Amendments

A Final Office Action was issued on March 1, 2002.

Appellant filed a Notice of Appeal on August 1, 2002 in response to the Final Office Action.

No amendments have been filed subsequent to the Final Office Action of March 1, 2002.

(5) Summary of Invention

Methods, systems, and computer-readable storage media are described that monitor memory usage, as summarized in the Specification on page 7, line 4 through page 11, line 18. Multiple usage thresholds (100, 106, 112) are established, and different actions (102, 104, 108, 110, 114) are taken as increasingly critical memory usage thresholds are reached. At each threshold, the objective is to free memory so that the next higher threshold is avoided.

At one threshold (112), one or more application programs (45) are simply requested to reduce their memory usage (114). The request is issued, for example, to the least recently active programs through their Windows® message loops. The applications (45) can respond to the requests as they deem appropriate. Well-behaved applications (45) will take steps to release resources as much as possible.

At another more critical threshold (106), the operating system closes one or more of the application programs (110). The application is closed using a standard operating system mechanism that allows the application (45) to shut down in an orderly fashion, while saving files and performing any other necessary housekeeping.

At another even more critical threshold (100), the operating system simply terminates one or more of the application programs (104). The application program's thread(s) are destroyed and all resources previously used by the application program (45) are released. The application program (45) is given no

opportunity to clean up, to save files, or to perform any other housekeeping chores.

(6) Issues

1. Whether claims 1, 7, 8, 9, 23, 29, and 30 are unpatentable over Matsumoto in view of Kubo when neither reference nor a combination of the references discloses or suggests the aspect of an operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over one or more application programs to reduce memory usage.

2. Whether claims 2, 13, 24, 32, 33, 34, 35, 36, 37, 38, and 39 are unpatentable over Matsumoto in view of Kubo when neither reference nor a combination of the references discloses or suggests the aspect of communicating a request to at least one application program for the at least one application program to limit its use of memory.

3. Whether claims 10, 11, 12, 14, 15, and 16 are unpatentable over Matsumoto in view of Kubo when neither reference nor a combination of the references discloses or suggests the aspect of at a first memory usage threshold requesting at least one of the application programs to close itself, and at a second memory usage threshold that is more critical than the first memory usage threshold terminating at least one of the application programs without allowing its further execution.

4. Whether claims 5, 6, 17, 18, 19, 22, 27, 28, and 31 are unpatentable over Matsumoto in view of Kubo when neither reference nor a combination of the references discloses or suggests the aspects of at a first memory threshold

requesting at least one of the application programs to limit its use of memory, at a second memory threshold requesting at least one of the application programs to close itself, and at a third memory threshold terminating at least one of the application programs without allowing its further execution.

5. Whether claim 40 is unpatentable over Kannan in view of Bishop when neither reference nor a combination of the references discloses or suggests the aspect of an application program being responsive to a particular message received through its message loop to reduce its current use of memory.

(7) Grouping of Claims

All of the claims do not stand or fall together. The claims are grouped as follows:

Group I: Claims 1, 3, 4, 7, 8, 9, 23, 25, 26, 29, and 30.

Group II: Claims 2, 13, 24, 32, 33, 34, 35, 36, 37, 38, and 39.

Group III: Claims 10, 11, 12, 14, 15, and 16.

Group IV: Claims 5, 6, 17, 18, 19, 20, 21, 22, 27, 28, and 31.

Group V: Claim 40.

(8) Argument

1. Group I: Claims 1, 7, 8, 9, 23, 29, and 30 are not taught or suggested by Matsumoto in view of Kubo because neither Matsumoto nor Kubo, nor the combination of Matsumoto and Kubo, discloses or suggests an operating system wielding, at increasingly critical memory thresholds,

correspondingly increasing control over one or more application programs to reduce memory usage.

In the Final Office Action dated March 1, 2002, claims 1, 7, 8, 9, 23, 29, and 30 (Group I) were rejected under 35 U.S.C. 103(a) as being unpatentable over Matsumoto in view of Kubo.

Matsumoto is directed to a computer operation management system for a computer operating system capable of simultaneously executing multiple application programs (see, Abstract). The underlying operating system in Matsumoto has a virtual storage area with a finite size – when a program is executed exceeding this finite size the program ends immediately (see, col. 9, lines 5-7). The management system of Matsumoto is situated between the operating system and the application programs, and operates to monitor and control execution of the application programs (see, Fig. 1 and col. 8, lines 50-60). Rather than having the operating system terminate an application, the management system of Matsumoto monitors resource usage and initiates its own recovery process if a threshold is exceeded (see, col. 19, lines 52-57; and col. 16, lines 29-32).

The management system of Matsumoto includes a resource manager that checks an amount of memory used and then compares the actual memory used with a control limit read from a program definition file (see, col. 16, lines 29-32). If the control limit is exceeded, an error recovery processor is notified (see, col. 16, lines 32-35). The error recovery process executes a predefined error recovery process also stored in the program definition file (see, col. 10, lines 37-40). The error recovery process comprises what measures to implement when an error

occurs, the recovery procedure, alarm notification, mail, and what measures to implement for any related programs (see, col. 15, lines 10-14).

Kubo is directed to scheduling a job in a clustered computer system (see, Abstract). In Kubo, a job selector selects jobs for execution by one of multiple different clusters (see, col. 3, lines 8-18). A measurement mechanism exists in each cluster to measure the resource utilization in the cluster after the previous measurement (see, col. 3, lines 22-26). A request can be sent to the job selector to start scheduling a new job for the cluster at two different times: one is when the measurement is completed (see, Fig. 3), and the other is when a job is completed (see, Fig. 4). In the first situation, when the measurement is completed, if the resource utilization is low then the request controller requests the job selector to start a new job in the cluster; however, if the resource utilization is not low, then a new job is not requested (see, col. 3, lines 41-52). In the second situation, when execution of a job is completed, the request controller judges whether the resource utilization of the cluster that completes execution is high (see, col. 3, line 65 – col. 4, line 3). If the utilization in the cluster is not high, then a request is sent to the job selector to start scheduling a new job for the cluster; however, if the utilization in the cluster is high, then such job selection is not required (see, col. 4, lines 4-9). Thus, although Kubo teaches two different trigger points (utilization high and utilization low) for determining whether to request a job be scheduled, which of these trigger points is used is based on whether a resource utilization measurement was just completed or execution of a job was just completed.

Additionally, when execution of a job is completed, the request controller judges whether the resource utilization is high based on a first threshold value that

is set to a value near a target resource utilization rate and a second threshold value is set to a value larger than the first threshold value (see, col. 4, lines 16-20). If the measured value exceeds the second threshold value, then the resource utilization is judged to be high, and if not, then the utilization is judged not to be high (see, col. 4, lines 20-23).

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. See, emphasis added, MPEP § 2142.

Appellant respectfully submits that it would not have been obvious to one of ordinary skill in the art to combine Matsumoto and Kubo. As discussed above, Matsumoto is directed to monitoring execution of application programs on a computer and invoking an error recovery process if an amount of memory used exceeds a control limit. Kubo, on the other hand, is directed to scheduling execution jobs across multiple different processor clusters. Appellant respectfully submits that there is no teaching or suggestion in the prior art to combine the monitoring and invoking of an error recovery process of Matsumoto with the job scheduling of Kubo.

However, assuming for the sake of argument that the Matsumoto and Kubo references are combined, Appellant respectfully submits that the combination does not disclose or suggest the aspect of an operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over one or more application programs to reduce memory usage. Claim 1, for example, recites:

> setting a plurality of memory thresholds; and
> the operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over said one or more application programs to reduce memory usage.

Kubo simply discloses determining whether a cluster is ready to have a new job scheduled for it. Which of two different resource utilization levels are used to make the determination are based on the event that caused the determination to occur. One such event is the resource utilization measurement that occurs at predetermined times (see, col. 3, lines 22-26 and 41-52). The other such event is job execution completion (see, col. 3, line 65 – col. 4, line 3). Appellant respectfully submits that these two events of Kubo are not increasingly critical memory thresholds. There is no discussion or suggestion in Kubo of an "increasingly critical" nature to these events – they are simply two different events that can be used to determine whether a cluster is ready to have a new job scheduled for it.

Kubo also discloses that, in determining whether a cluster is ready to have a new job scheduled for it due to completing a previous job execution, the determining is based on whether its second threshold is exceeded (see, col. 4, lines 20-23). This decision is based on the second threshold, not on the first threshold (the first threshold appears to simply be a reference point for determining what the second threshold should be). Whether the measured resource utilization exceeds

or does not exceed the first threshold is irrelevant – Kubo discloses no comparison of any values to this first threshold in order to determine whether job scheduling is needed. Thus, there is no concept in Kubo of increasing control at increasingly critical thresholds. There are only two possible results in Kubo when completing a previous job execution – a job is selected for execution by the cluster or a job is not selected for execution by the cluster. Which of these results occurs is based solely on the second threshold – there is nothing about increasing the control over the cluster for the different thresholds.

Thus, Appellant respectfully submits that Kubo does not disclose or suggest the aspect of the operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over said one or more application programs to reduce memory usage.

Furthermore, Appellant respectfully submits that Matsumoto does not cure this deficiency of Kubo. In Matsumoto, a computer operation management system including a computer resource manager exists between the operating system and the applications (see, Figs. 1 and 2). This allows the management system to monitor resource usage and initiate its own recovery process if the threshold is exceeded, rather than having the operating system immediately terminate the application and generate an error (see, col. 19, lines 52-57; and col. 16, lines 29-32). Thus, notification of the error recovery means is the only action taken because the management system is able to intervene before the operating system would have to terminate (abnormally end) the application – the abnormal endings are thus prevented (see, col. 19, lines 52-57).

Thus, Appellant respectfully submits that Matsumoto does not disclose the operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over said one or more application programs to reduce memory usage. Matsumoto's goal is to replace one level of control (termination) with another level of control (notice), not to have the two levels coexist. Furthermore, Matsumoto discloses two different entities that are providing the level of control – the management system provides the notice level while the underlying operating system provides the termination level. Appellant respectfully submits that there is no disclosure or suggestion in Matsumoto of a single entity providing both of these levels, and thus no disclosure or suggestion of an operating system wielding increasing control as claimed for example in claim 1.

Thus, given that neither Matsumoto nor Kubo discloses or suggests the aspect of an operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over one or more application programs to reduce memory usage, Appellant respectfully submits that the combination of Matsumoto and Kubo does not disclose or suggest the aspect of an operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over one or more application programs to reduce memory usage.

Accordingly, Appellant respectfully submits that claims 1, 7, 8, 9, 23, 29, and 30 are allowable over the cited references and that the rejection should be withdrawn.

2. Group II: Claims 2, 13, 24, 32, 33, 34, 35, 36, 37, 38, and 39 are not taught or suggested by Matsumoto in view of Kubo because neither Matsumoto nor Kubo, nor the combination of Matsumoto and Kubo, discloses or suggests communicating a request to at least one application program for the at least one application program to limit its use of memory.

In the Final Office Action dated March 1, 2002, claims 2, 13, 24, 32, 33, 34, 35, 36, 37, 38, and 39 (Group II) were rejected under 35 U.S.C. 103(a) as being unpatentable over Matsumoto in view of Kubo.

As discussed above, Matsumoto is directed to a computer operation management system for a computer operating system capable of simultaneously executing multiple application programs (see, Abstract). The underlying operating system in Matsumoto has a virtual storage area with a finite size – when a program is executed exceeding this finite size the program ends immediately (see, col. 9, lines 5-7). The management system of Matsumoto is situated between the operating system and the application programs, and operates to monitor and control execution of the application programs (see, Fig. 1 and col. 8, lines 50-60). Rather than having the operating system terminate an application, the management system of Matsumoto monitors resource usage and initiates its own recovery process if a threshold is exceeded (see, col. 19, lines 52-57; and col. 16, lines 29-32).

Also as discussed above, Kubo is directed to scheduling a job in a clustered computer system (see, Abstract). In Kubo, a job selector selects jobs for execution by one of multiple different clusters (see, col. 3, lines 8-18). A measurement mechanism exists in each cluster to measure the resource utilization in the cluster

after the previous measurement (see, col. 3, lines 22-26). A request can be sent to the job selector to start scheduling a new job for the cluster at two different times: one is when the measurement is completed (see, Fig. 3), and the other is when a job is completed (see, Fig. 4).

Neither Matsumoto nor Kubo is relied on in the Final Office Action dated March 1, 2002 as disclosing or suggesting the aspect of communicating a request to at least one application program for the at least one application program to limit its use of memory. Furthermore, Appellant respectfully submits that nothing in the management system of Matsumoto nor the job scheduling of Kubo discloses or suggests the aspect of communicating a request to at least one application program for the at least one application program to limit its use of memory.

In the Final Office Action dated March 1, 2002, with respect to claim 2 for example, it was asserted that "limiting, closing, or terminating of a program are well known in the art (MPEP2144.03)" (see, ¶4, p. 3). Appellant respectfully traverses this rejection and submits that "communicating a request to at least one of the application programs for the at least one application program to limit its use of memory" as claimed in claim 2 is not well known in the art. In response to Appellant's previous request (in Appellant's Response filed December 6, 2001) that the rejection to claim 2 be withdrawn or evidence cited teaching "communicating a request to at least one of the application programs for the at least one application program to limit its use of memory" as claimed in claim 2, Applicant was "directed to the references cited in the 6 October 1999 Office Action (i.e., Bishop, Matsumoto, Kannan, and Jewett)." It was further asserted that "the Microsoft Windows User's Guide Version 3.0 is provided to show

additional teachings for controlling program operations and freeing up memory". See, March 1, 2002 Final Office Action, page 6.

In the October 6, 1999 Office Action, U.S. Patent No. 5,826,082 to Bishop et al. (hereinafter "Bishop") was relied on as teaching:

> . . . at a less critical memory threshold (resource manager determines in decision block 204, c4 l52-62) interacting with at least one of the application programs to limit its use of memory (suspend a prior request, Id.).

Applicant respectfully submits that Bishop does not disclose or suggest "at a less critical memory threshold, communicating a request to at least one of the application programs for the at least one application program to limit its use of memory" as claimed in claim 2.

Bishop discloses a computer system including a resource manager that is responsible for allocation of the computer system's resources (see, col. 2, lines 35-37). A thread that needs a resource submits a request to the resource manager for the necessary resource including a requested amount of the resource (see, col. 3, lines 53-62). If the requested amount is not available, then the resource manager searches for a prior request(s) of a separate thread that has already reserved enough of the resource to satisfy the request, and attempts to suspend the prior request(s) (see, col. 4, lines 52-57 and 63-67; and col. 5, lines 1-4 and 23-31). This suspending of the prior request is transparent to the user of the computer system (see, col. 5, lines 30-31).

In contrast, claim 2 recites "communicating a request to at least one of the application programs for the at least one application program to limit its use of memory". Bishop discloses the resource manager suspending a prior request for resources, not communicating a request to an application program for the

application program itself to limit its use of memory as claimed in claim 2. Applicant respectfully submits that there is no discussion or suggestion whatsoever in Bishop of communicating a request to an application program for the application program itself to limit its use of memory as claimed in claim 2.

With respect to the Microsoft Windows User's Guide Version 3.0 (hereinafter, the "User's Guide"), the User's Guide discloses ways of "Freeing Up Memory". The User's Guide instructs users that they can free up memory in a number of ways, such as closing other applications, running non-windows applications in a full-screen display instead of in a window, minimizing Windows applications to icons, clearing or saving the clipboard contents, setting the desktop wallpaper to None, changing PIF (Program Information File) options for an application (such as reducing the amount specified in "Memory Requirements: KB Required", making sure that "XMS Memory: KB Required" and "XMS Memory: KB Limit" are set to 0) (see, pages 478-479).

Applicant respectfully submits, however, that nowhere in these instructions of how a user can free up memory is there any disclosure or suggestion of "communicating a request to at least one of the application programs for the at least one application program to limit its use of memory" as claimed in claim 2. Applicant respectfully submits that a user closing an application, minimizing Windows applications to icons, changing PIF options for an application, and so forth are ways for a user to dictate a change in memory usage for an application, but are not an application program being requested to limit its use of memory as claimed in claim 2.

Thus, given that none of Matsumoto, Kubo, Bishop, or the User's Guide discloses or suggests the aspect of communicating a request to at least one application program for the at least one application program to limit its use of memory, Applicant respectfully submits that any combination of Matsumoto, Kubo, Bishop, and the User's Guide does not disclose or suggest the aspect of communicating a request to at least one application program for the at least one application program to limit its use of memory.

Accordingly, Appellant respectfully submits that claims 2, 13, 24, 32, 33, 34, 35, 36, 37, 38, and 39 are allowable over the cited references and that the rejection should be withdrawn.

3. Group III: Claims 10, 11, 12, 14, 15, and 16 are not taught or suggested by Matsumoto in view of Kubo because neither Matsumoto nor Kubo, nor the combination of Matsumoto and Kubo, discloses or suggests at a first memory usage threshold requesting at least one of the application programs to close itself, and at a second memory usage threshold that is more critical than the first memory usage threshold terminating at least one of the application programs without allowing its further execution.

In the Final Office Action dated March 1, 2002, claims 10, 11, 12, 14, 15, and 16 (Group III) were rejected under 35 U.S.C. 103(a) as being unpatentable over Matsumoto in view of Kubo.

As discussed above, Matsumoto is directed to a computer operation management system for a computer operating system capable of simultaneously executing multiple application programs (see, Abstract). The underlying operating

system in Matsumoto has a virtual storage area with a finite size – when a program is executed exceeding this finite size the program ends immediately (see, col. 9, lines 5-7). The management system of Matsumoto is situated between the operating system and the application programs, and operates to monitor and control execution of the application programs (see, Fig. 1 and col. 8, lines 50-60). Rather than having the operating system terminate an application, the management system of Matsumoto monitors resource usage and initiates its own recovery process if a threshold is exceeded (see, col. 19, lines 52-57; and col. 16, lines 29-32).

Also as discussed above, Kubo is directed to scheduling a job in a clustered computer system (see, Abstract). In Kubo, a job selector selects jobs for execution by one of multiple different clusters (see, col. 3, lines 8-18). A measurement mechanism exists in each cluster to measure the resource utilization in the cluster after the previous measurement (see, col. 3, lines 22-26). A request can be sent to the job selector to start scheduling a new job for the cluster at two different times: one is when the measurement is completed (see, Fig. 3), and the other is when a job is completed (see, Fig. 4).

Neither Matsumoto nor Kubo is relied on in the Final Office Action dated March 1, 2002 as disclosing or suggesting two such memory thresholds and the actions taken at each, as recited for example in claim 10. Furthermore, Applicant respectfully submits that nothing in the management system of Matsumoto nor the job scheduling of Kubo discloses or suggests two such memory thresholds.

In the Final Office Action dated March 1, 2002, it was asserted that "limiting, closing, or terminating of a program are well known in the art

(MPEP2144.03)" (see, ¶4, p. 3). Applicant respectfully traverses this rejection and submits that at a first memory usage threshold requesting at least one of the application programs to close itself, and at a second memory usage threshold that is more critical than the first memory usage threshold terminating at least one of the application programs without allowing its further execution is not well known in the art. In response to Applicant's previous request (in Applicant's Response filed December 6, 2001) that the rejection be withdrawn or evidence cited teaching the elements, Applicant was "directed to the references cited in the 6 October 1999 Office Action (i.e., Bishop, Matsumoto, Kannan, and Jewett)." It was further asserted that "the Microsoft Windows User's Guide Version 3.0 is provided to show additional teachings for controlling program operations and freeing up memory". See, March 1, 2002 Final Office Action, page 6.

In the October 6, 1999 Office Action, Jewett was relied on as teaching:

> Jewett teaches (processes ... perform some cleanup activity as required for the particular application, c25 l3-11) which corresponds to at a second memory threshold, requesting at least one of the application programs to close itself.

With respect to claim 10 for example, Applicant respectfully submits that Jewett and claim 10 are directed to nonanalogous arts. In order to rely on a reference under 35 U.S.C. §103, the reference must be analogous prior art. See, MPEP § 2141.01(a). Claim 10 is directed to controlling memory usage in a computer system having limited physical memory, whereas Jewett is directed to a shutdown and restart procedure in the event of a power failure (see, col. 1, lines 25-28; and col. 2, line 45 – col. 3, line 9). Thus, Applicant respectfully submits that Jewett is not a valid §103 reference for rejecting claim 10.

However, assuming for the sake of argument that Jewett and claim 10 are directed to analogous arts, Applicant respectfully submits that Jewett does not disclose or suggest "at a first memory threshold, requesting at least one of the application programs to close itself" as claimed in claim 10. Jewett discloses that a power failure is sensed and a shutdown time period provided in which active processes will be given a warning of the impending shutdown so that they can perform any preparations necessary (see, col. 22, lines 38-65). Applicant respectfully submits that allowing a process to prepare for an impending shutdown due to a power failure does not disclose or suggest at a first memory threshold requesting at least one of the application programs to close itself as claimed in claim 10.

With respect to the Microsoft Windows User's Guide Version 3.0 (the "User's Guide"), the User's Guide discloses ways of "Freeing Up Memory". The User's Guide instructs users that they can free up memory in a number of ways, such as closing other applications, running non-windows applications in a full-screen display instead of in a window, minimizing Windows applications to icons, clearing or saving the clipboard contents, setting the desktop wallpaper to None, changing PIF (Program Information File) options for an application (such as reducing the amount specified in "Memory Requirements: KB Required", making sure that "XMS Memory: KB Required" and "XMS Memory: KB Limit" are set to 0) (see, pages 478-479).

Applicant respectfully submits that the cited references do not disclose or suggest the two thresholds of claim 10. Applicant respectfully submits that instructing a user on how the user can free up memory does not disclose or suggest

two separate thresholds and what actions are taken at each of those thresholds. As further discussed above, Jewett does not cure these deficiencies of Matsumoto and Kubo, and thus, the combination of the cited references does not disclose or suggest the two separate thresholds and what actions are taken at each of those thresholds as recited in claim 10.

Accordingly, Appellant respectfully submits that claims 10, 11, 12, 14, 15, and 16 are allowable over the cited references and that the rejection should be withdrawn.

4. Group IV: Claims 5, 6, 17, 18, 19, 22, 27, 28, and 31 are not taught or suggested by Matsumoto in view of Kubo because neither Matsumoto nor Kubo, nor the combination of Matsumoto and Kubo, discloses or suggests at a first memory threshold requesting at least one of the application programs to limit its use of memory, at a second memory threshold requesting at least one of the application programs to close itself, and at a third memory threshold terminating at least one of the application programs without allowing its further execution.

In the Final Office Action dated March 1, 2002, claims 5, 6, 17, 18, 19, 22, 27, 28, and 31 (Group IV) were rejected under 35 U.S.C. 103(a) as being unpatentable over Matsumoto in view of Kubo.

As discussed above, Matsumoto is directed to a computer operation management system for a computer operating system capable of simultaneously executing multiple application programs (see, Abstract). The underlying operating system in Matsumoto has a virtual storage area with a finite size – when a program

is executed exceeding this finite size the program ends immediately (see, col. 9, lines 5-7). The management system of Matsumoto is situated between the operating system and the application programs, and operates to monitor and control execution of the application programs (see, Fig. 1 and col. 8, lines 50-60). Rather than having the operating system terminate an application, the management system of Matsumoto monitors resource usage and initiates its own recovery process if a threshold is exceeded (see, col. 19, lines 52-57; and col. 16, lines 29-32).

Also as discussed above, Kubo is directed to scheduling a job in a clustered computer system (see, Abstract). In Kubo, a job selector selects jobs for execution by one of multiple different clusters (see, col. 3, lines 8-18). A measurement mechanism exists in each cluster to measure the resource utilization in the cluster after the previous measurement (see, col. 3, lines 22-26). A request can be sent to the job selector to start scheduling a new job for the cluster at two different times: one is when the measurement is completed (see, Fig. 3), and the other is when a job is completed (see, Fig. 4).

Neither Matsumoto nor Kubo is relied on in the Final Office Action dated March 1, 2002 as disclosing or suggesting three such memory thresholds. Furthermore, Applicant respectfully submits that nothing in the management system of Matsumoto nor the job scheduling of Kubo discloses or suggests three such memory thresholds. Additionally, as discussed above with respect to Group I, Kubo discloses deciding one of two possible outcomes (a job is selected for execution or a job is not selected for execution), not three possible outcomes based on three thresholds.

In the Final Office Action dated March 1, 2002, with respect to claim 5 for example, it was asserted that "limiting, closing, or terminating of a program are well known in the art (MPEP2144.03)" (see, ¶4, p. 3). Applicant respectfully traverses this rejection and submits that at a first memory threshold requesting at least one of the application programs to limit its use of memory, at a second memory threshold requesting at least one of the application programs to close itself, and at a third memory threshold terminating at least one of the application programs without allowing its further execution is not well known in the art. In response to Applicant's previous request (in Applicant's Response filed December 6, 2001) that the rejection to claim 5 be withdrawn or evidence cited teaching the prompting of claim 5, Applicant was "directed to the references cited in the 6 October 1999 Office Action (i.e., Bishop, Matsumoto, Kannan, and Jewett)." It was further asserted that "the Microsoft Windows User's Guide Version 3.0 is provided to show additional teachings for controlling program operations and freeing up memory". See, March 1, 2002 Final Office Action, page 6.

Applicant respectfully submits that the cited references do not disclose or suggest such a three-threshold process as claimed in claim 5. Analogous to the discussion above in Group II, Applicant respectfully submits that the cited references, including Bishop, do not disclose or suggest at a first memory threshold, requesting at least one of the application programs to limit its use of memory as recited in claim 5.

Additionally, in the October 6, 1999 Office Action, Jewett was relied on as teaching:

> Jewett teaches (processes ... perform some cleanup activity as required for the particular application, c25 l3-11) which corresponds

to at a second memory threshold, requesting at least one of the application programs to close itself.

With respect to claim 5 for example, Applicant respectfully submits that Jewett and claim 5 are directed to nonanalogous arts. In order to rely on a reference under 35 U.S.C. §103, the reference must be analogous prior art. See, MPEP § 2141.01(a). Claim 5 is directed to a method of controlling memory usage in a computer system having limited physical memory, whereas Jewett is directed to a shutdown and restart procedure in the event of a power failure (see, col. 1, lines 25-28; and col. 2, line 45 – col. 3, line 9). Thus, Applicant respectfully submits that Jewett is not a valid §103 reference for rejecting claim 5.

However, assuming for the sake of argument that Jewett and claim 5 are directed to analogous arts, Applicant respectfully submits that Jewett does not disclose or suggest "at a second memory threshold, requesting at least one of the application programs to close itself" as claimed in claim 5. Jewett discloses that a power failure is sensed and a shutdown time period provided in which active processes will be given a warning of the impending shutdown so that they can perform any preparations necessary (see, col. 22, lines 38-65). Applicant respectfully submits that allowing a process to prepare for an impending shutdown due to a power failure does not disclose or suggest at a second memory threshold requesting at least one of the application programs to close itself as claimed in claim 5.

With respect to the Microsoft Windows User's Guide Version 3.0 (the "User's Guide"), the User's Guide discloses ways of "Freeing Up Memory". The User's Guide instructs users that they can free up memory in a number of ways, such as closing other applications, running non-windows applications in a full-

screen display instead of in a window, minimizing Windows applications to icons, clearing or saving the clipboard contents, setting the desktop wallpaper to None, changing PIF (Program Information File) options for an application (such as reducing the amount specified in "Memory Requirements: KB Required", making sure that "XMS Memory: KB Required" and "XMS Memory: KB Limit" are set to 0) (see, pages 478-479).

Applicant respectfully submits, however, that nowhere in these instructions of how a user can free up memory is there any disclosure or suggestion of requesting at least one of the application programs to limit its use of memory, analogous to the discussion above regarding Group II.

Given that none of the cited references, individually or in combination, discloses or suggests the element of requesting at least one of the application programs to limit its use of memory, Applicant respectfully submits that the cited references do not disclose or suggest the three thresholds including at a first memory threshold, requesting at least one of the application programs to limit its use of memory as recited in claim 5.

Furthermore, Applicant respectfully submits that the cited references do not disclose or suggest the operating system wielding increasing control comprises the three thresholds as recited in claim 5. As discussed above with respect to Group I, Matsumoto in view of Kubo does not disclose or suggest the operating system wielding increasing control, much less of three separate thresholds and what actions are taken at each of those thresholds. As further discussed above, neither Bishop nor Jewett cures these deficiencies of Matsumoto and Kubo, and thus, the combination of the cited references does not disclose or suggest the three separate

thresholds and what actions are taken at each of those thresholds as recited in claim 5.

Accordingly, Appellant respectfully submits that claims 5, 6, 17, 18, 19, 22, 27, 28, and 31 are allowable over the cited references and that the rejection should be withdrawn.

5. Group V: Claim 40 is not taught or suggested by Kannan in view of Bishop because neither Kannan nor Bishop, nor the combination of Kannan and Bishop, discloses or suggests an application program being responsive to a particular message received through its message loop to reduce its current use of memory.

In the Final Office Action dated March 1, 2002, claim 40 (Group V) was rejected under 35 U.S.C. 103(a) as being unpatentable over Kannan in view of Bishop.

Kannan discloses a system in which an exception handler intercepts exceptions that are fatal and that would normally cause an application to be terminated (see, col. 4, lines 44-47). The exception handler notifies a crash guard process of the fatal exception, which in turn interacts with the user to determine whether to continue executing the application or to terminate it (see, col. 4, lines 47-51). The crash guard process displays a warning dialog that allows the user to choose between continuing to work and terminating the application (see, Fig. 4, and col. 7, lines 34-47). Thus, the system of Kannan allows a user to continue using an application that has generated a fatal exception that would otherwise have

caused the operating system to terminate execution of the application (see, col. 2, lines 39-43).

Bishop, on the other hand, discloses a computer system including a resource manager that is responsible for allocation of the computer system's resources (see, col. 2, lines 35-37). A thread that needs a resource submits a request to the resource manager for the necessary resource including a requested amount of the resource (see, col. 3, lines 53-62). If the requested amount is not available, then the resource manager searches for a prior request(s) of a separate thread that has already reserved enough of the resource to satisfy the request, and attempts to suspend the prior request(s) (see, col. 4, lines 52-57 and 63-67; and col. 5, lines 1-4 and 23-31). This suspending of the prior request is transparent to the user of the computer system (see, col. 5, lines 30-31).

Applicant respectfully submits that it would not have been obvious to one of ordinary skill in the art to combine Kannan and Bishop. As discussed above, Kannan is directed to allowing a user to continue using an application that has generated a fatal exception that would otherwise have caused the operating system to terminate execution of the application, whereas Bishop is directed to reserving resources and suspending prior requests transparently to the user. Applicant respectfully submits that there is no teaching or suggestion to combine the allowing a user to select between continuing and terminating an application that has generated a fatal exception of Kannan with the resource management and user-transparent resource request suspension of Bishop.

However, assuming for the sake of argument that the Kannan and Bishop references are combined, Applicant respectfully submits that the combination does

not disclose or suggest an application program that resides in a computer-readable memory for execution by a processor as recited in claim 40. Claim 40 recites:

> An application program that resides in a computer-readable memory for execution by a processor in conjunction with an operating system, the application program having a message loop that receives messages from an operating system, the application program being responsive to a particular message received through its message loop to reduce its current use of memory.

Kannan is not cited as disclosing, and Applicant respectfully submits that Kannan does not disclose or suggest, an application program being responsive to a particular message received through its message loop to reduce its current use of memory as claimed in claim 40.

Bishop, on the other hand, is cited as disclosing an application operation that is programmed to reduce its current use of memory (see, ¶5, p. 5). Applicant respectfully disagrees with this assertion. Bishop discloses a resource manager that is responsible for allocation of the computer's resources and that has the ability to suspend requests for resources (see, col. 5, lines 26-31). The resource manager determines whether to suspend a prior request of a thread based on the priorities of the current and prior requests, or the priorities of the current and prior requesting threads (see, col. 5, lines 17-22). Neither the current nor the prior requesting thread determines whether its request is to be suspended – the resource manager makes the determination. Thus, since neither of the threads makes this determination, neither of the threads itself is programmed (nor is the program the threads are part of programmed) to reduce its current use of memory (rather, the resource manager forces one of their requests to be suspended, and thus forces a suspension in the use of resources by the thread whose request is suspended).

Thus, Applicant respectfully submits that Bishop does not cure the deficiencies of Kannan. Given that neither Bishop nor Kannan discloses or suggests the aspect of an application program being responsive to a particular message received through its message loop to reduce its current use of memory, Applicant respectfully submits that the combination of Bishop and Kannan does not disclose or suggest the aspect of an application program being responsive to a particular message received through its message loop to reduce its current use of memory.

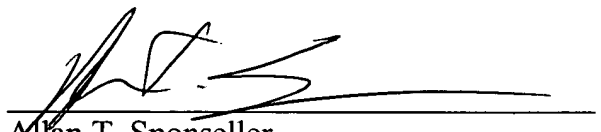
Accordingly, Appellant respectfully submits that claim 40 is allowable over the cited references and that the rejection should be withdrawn.

Conclusion

The Office's basis and supporting rationale for the § 103(a) rejections is not supported by the teaching of the cited references. Applicant respectfully requests that the rejections be overturned and that pending claims 1-40 be allowed to issue.

Respectfully Submitted,

Dated: 11/1/02

By: 

Allan T. Sponseller
Lee & Hayes, PLLC
Reg. No. 38,318
(509) 324-9256 ext. 215

(9) Appendix of Appealed Claims

1. A method of controlling memory usage in a computer system having limited physical memory, wherein one or more application programs execute in conjunction with an operating system, the method comprising:

setting a plurality of memory thresholds; and

the operating system wielding, at increasingly critical memory thresholds, correspondingly increasing control over said one or more application programs to reduce memory usage.

2. A method as recited in claim 1, wherein the wielding increasing operating system control comprises:

at a less critical memory threshold, communicating a request to at least one of the application programs for the at least one application program to limit its use of memory; and

at a more critical memory threshold, terminating at least one of the application programs without allowing its further execution.

3. A method as recited in claim 1, wherein the wielding increasing operating system control comprises:

prompting a user to select at least one of the application programs and then the operating system requesting that the at least one selected application program close itself.

4. A method as recited in claim 1, wherein the wielding increasing operating system control comprises:

prompting a user to select at least one of the application programs and then terminating it without allowing its further execution.

5. A method as recited in claim 1, wherein the wielding increasing operating system control comprises:

at a first memory threshold, requesting at least one of the application programs to limit its use of memory;

at a second memory threshold, requesting at least one of the application programs to close itself; and

at a third memory threshold, terminating at least one of the application programs without allowing its further execution.

6. A method as recited in claim 1, wherein the wielding increasing operating system control comprises:

at a first memory threshold, requesting at least one of the application programs to limit its use of memory;

at a second memory threshold, prompting a user to select at least one of the application programs and then requesting it to close itself; and

at a third memory threshold, prompting the user to select at least one of the application programs and then terminating it without allowing its further execution.

7. A method as recited in claim 1, further comprising:

at one or more of the memory thresholds, reclaiming unused stack memory.

8. A method as recited in claim 1, further comprising:

at one or more of the memory thresholds, discarding read-only memory.

9. A computer-readable storage medium having computer-executable instructions for performing the method recited in claim 1.

10. A computer-readable storage medium having instructions for controlling memory usage in a computer system having limited physical memory, wherein one or more application programs execute in conjunction with an operating system, the instructions being executable by the computer system to perform acts comprising:

at a first memory usage threshold, requesting at least one of the application programs to close itself; and

at a second memory usage threshold that is more critical than the first memory usage threshold, terminating at least one of the application programs without allowing its further execution.

11. A computer-readable storage medium as recited in claim 10, the instructions being executable to perform additional acts comprising:

before performing the requesting step, prompting a user to select one of the application programs to be closed; and

before performing the terminating step, prompting the user to select one of the application programs to be terminated.

12. A computer-readable storage medium as recited in claim 10, the instructions being executable to perform additional acts comprising:

before performing the requesting step, requiring a user to select one of the application programs to be closed; and

before performing the terminating step, requiring the user to select one of the application programs to be terminated.

13. A computer-readable storage medium as recited in claim 10, the instructions being executable to perform an additional act comprising:

at a further memory threshold that is less critical than the first and second memory usage thresholds, requesting at least one of the application programs to limit its use of memory.

14. A computer-readable storage medium as recited in claim 10, the instructions being executable to perform an additional act comprising:

reclaiming unused stack memory before requesting at least one of the application programs to close itself and before terminating at least one of the application programs.

15. A computer-readable storage medium as recited in claim 10, the instructions being executable to perform an additional act comprising:

discarding read-only memory before requesting at least one of the application programs to close itself and before terminating at least one of the application programs.

16. A computer-readable storage medium as recited in claim 10, the instructions being executable to perform additional acts comprising:

reclaiming unused stack memory and discarding read-only memory before requesting at least one of the application programs to close itself and before terminating at least one of the application programs.

17. A method of controlling memory usage in a computer system having limited physical memory, wherein one or more application programs execute in conjunction with an operating system, the method comprising:

at a first memory usage threshold, requesting at least one of the application programs to limit its use of memory;

at a second memory usage threshold that is more critical than the first memory usage threshold, requesting at least one of the application programs to close itself;

at a third memory usage threshold that is more critical than the first and second memory usage thresholds, terminating at least one of the application programs without allowing its further execution; and

reclaiming unused stack memory and discarding read-only memory before requesting at least one of the application programs to close itself and before terminating at least one of the application programs.

18. A method as recited in claim 17, wherein the reclaiming and discarding are performed at further memory usage thresholds that are set in relation to the second and third memory usage thresholds.

19. A method as recited in claim 17, wherein the reclaiming and discarding are performed at further memory usage thresholds that are set in relation to the first, second, and third memory usage thresholds.

20. A method as recited in claim 17, further comprising:

before performing the requesting, prompting a user to select one of the application programs to be closed; and

before performing the terminating, prompting the user to select one of the application programs to be terminated.

21. A method as recited in claim 17, further comprising:

before performing the requesting, requiring a user to select one of the application programs to be closed; and

before performing the terminating, requiring the user to select one of the application programs to be terminated.

22. A computer-readable storage medium having computer-executable instructions for performing the method recited in claim 17.

23. A computer system comprising:

a processor;

an operating system that is executable by the processor and that utilizes the physical memory;

a virtual memory system that includes physical memory but does not include secondary storage;

one or more application programs that utilize the virtual memory system;

wherein the operating system is configured to perform the following acts:

monitoring physical memory usage; and

at increasingly critical physical memory usage thresholds, wielding increasing control over said one or more application programs to reduce physical memory usage.

24. A computer system as recited in claim 23, wherein the act of wielding increasing control comprises the following acts:

at a less critical memory threshold, communicating a request to at least one of the application programs for the at least one application program to limit its use of memory; and

at a more critical memory threshold, terminating at least one of the application programs without allowing its further execution.

25. A computer system as recited in claim 23, wherein the act of wielding increasing control comprises the following act:

prompting a user to select at least one of the application programs and then the operating system requesting that the at least one selected application program close itself.

26. A computer system as recited in claim 23, wherein the act of wielding increasing control comprises the following act:

prompting a user to select at least one of the application programs and then terminating it without allowing its further execution.

27. A computer system as recited in claim 23, wherein the act of wielding increasing control comprises the following acts :

at a first memory threshold, requesting at least one of the application programs to limit its use of memory;

at a second memory threshold, requesting at least one of the application programs to close itself; and

at a third memory threshold, terminating at least one of the application programs without allowing its further execution.

28. A computer system as recited in claim 23, wherein the act of wielding increasing control comprises the following acts:

at a first memory threshold, requesting at least one of the application programs to limit its use of memory;

at a second memory threshold, prompting a user to select at least one of the application programs and then requesting it to close itself; and

at a third memory threshold, prompting the user to select at least one of the application programs and then terminating it without allowing its further execution.

29. A computer system as recited in claim 23, wherein the operating system is further configured to perform the following additional act:

at one or more of the memory thresholds, reclaiming unused stack memory.

30. A computer system as recited in claim 23, wherein the operating system is further configured to perform the following additional act:

at one or more of the memory thresholds, discarding read-only memory.

31. A computer system as recited in claim 23, wherein the act of wielding increasing control comprises the following acts:

at a first memory threshold, requesting at least one of the application programs to limit its use of memory;

at a second memory threshold, prompting a user to select at least one of the application programs and then requesting that the at least one selected application program close itself;

at a third memory threshold, prompting the user to select at least one of the application programs and then terminating it without allowing its further execution; and

before prompting the user, reclaiming unused stack memory and discarding read-only memory.

32. A method of controlling memory usage in a computer system having limited physical memory, wherein one or more application programs execute in conjunction with an operating system, the method comprising:

monitoring memory usage; and

when memory usage is high, sending a message from the operating system to at least one of the application programs requesting the application program to reduce its current use of memory.

33. A method as recited in claim 32, further comprising sending the message to the application program when memory usage reaches a defined threshold.

34. A method as recited in claim 32, wherein the application programs have respective message loops, the method further comprising sending the message to the application program through its message loop.

35. A method as recited in claim 32, wherein the application programs have respective message loops, the method further comprising sending the message to a particular application program that was least recently active.

36. A computer-readable storage medium having computer-executable instructions for performing the method recited in claim 32.

37. A computer-readable storage medium having instructions for controlling memory usage in a computer system having limited physical memory, wherein one or more application programs execute in conjunction with an operating system, the instructions being executable by the computer system to perform acts comprising:

monitoring memory usage; and

at a defined memory usage threshold, sending a message from the operating system to at least one of the application programs requesting the application program to reduce its current use of memory.

38. A computer-readable storage medium as recited in claim 37, wherein the application programs have respective message loops, the instructions being executable to perform a further act of sending the message to the application program through its message loop.

39. A computer-readable storage medium as recited in claim 37, wherein the application programs have respective message loops, the instructions being executable to perform a further act of sending the message to a particular application program that was least recently active.

40. An application program that resides in a computer-readable memory for execution by a processor in conjunction with an operating system, the application program having a message loop that receives messages from an

operating system, the application program being responsive to a particular message received through its message loop to reduce its current use of memory.